# Configuring RSI on the controller

This guide highlights the steps needed in order to successfully configure the **RSI interface** on the controller to work with the **kuka_rsi_hardware_interface** on your PC with ROS.

## 1. Controller network configuration

Windows runs behind the SmartHMI on the teach pad. Make sure that the **Windows interface** of the controller and the **PC with ROS** is connected to the same subnet.

1. Log in as **Expert** or **Administrator** on the teach pad and navigate to **Network configuration** (**Start-up > Network configuration > Activate advanced configuration**).
2. There should already be an interface checked out as the **Windows interface**. For example:
    - **IP**: 192.168.250.20
    - **Subnet mask**: 255.255.255.0
    - **Default gateway**: 192.168.250.20
    - **Windows interface checkbox** should be checked.
3. Minimize the SmartHMI (**Start-up > Service > Minimize HMI**).
4. Run **RSI-Network** from the Windows Start menu (**All Programs > RSI-Network**).
5. Check that the **Network - Kuka User Interface** show the Windows interface with the specified IP address.
6. Add a new IP address on another subnet (e.g. 192.168.1.20) for the **RSI interface**.
    - Select the entry **New** under **RSI Ethernet** in the tree structure and press **Edit**.
    - Enter the IP address and confirm with **OK**.
    - Close **RSI-Network** and maximize the SmartHMI.
7. Reboot the controller with a cold restart (**Shutdown > Check *Force cold start* and *Reload files* > Reboot control PC**).
8. After reboot, minimize the SmartHMI (**Start-up > Service > Minimize HMI**).
9. Run **cmd.exe** and ping the PC you want to communicate with on the same subnet (e.g. 192.168.250.xx).

If your **PC** has an IP address on the same subnet as the **Windows interface** on the controller, the controller should receive answers from the PC:

- If this is the case, add another IP address to the current PC connection (e.g. 192.168.1.xx) on the same subnet as the **RSI** interface.

## 2. KRL Files

The files included in this folder specifies the data transferred via RSI. Some of the files needs to be modified to work for your specific configuration.

**ros_rsi_ethernet.xml**

1. Edit the `IP_NUMBER` tag so that it corresponds to the IP address (192.168.1.xx) previously added for your PC.
2. Keep the `PORT` tag as it is (49152) or change it if you want to use another port.

Note that the `rsi/listen_address` and `rsi/listen_port` parameters of the `kuka_rsi_hw_interface` must correspond to the `IP_NUMBER`and `PORT` set in these KRL files.

#### ros_rsi.rsi.xml

This file may be edited with application specific joint limits in degrees.

- Edit the parameters within the RSIObject `AXISCORR` to specify joint limits such as **LowerLimA1**, **UpperLimA1** etc. Note that these limits are in reference to the start position of the robot.
- Edit the parameters within `AXISCORRMON` to specify the overall correction limitation. If this limit is exceeded in either of the joint directions, RSI is stopped. The values of **MaxA1**, **MaxA2** etc. may be large to allow free movement within the specified joint limits in `AXISCORR`.

Notice the RSIObject of type `ETHERNET`. Within this object is a parameter called **Timeout**. This parameter is set to **100** by default. The RSI interface operates at `250 Hz` (4ms cycle). The **kuka_rsi_hardware_interface** does not have a period configured and is completely driven by the controller's output. Every controller RSI output has a IPOC timestamp which increments for every cycle. The **kuka_rsi_hardware_interface** will answer as quickly as possible. The answer includes the last IPOC received. If the connected **PC with ROS** did not manage to answer within the RSI cycle of **4ms**, the IPOC timestamp of RSI has incremented. The IPOC included in the answer is not matched and the controller will increment a counter. When this counter hits the **Timeout** parameter (**100**), the RSI connection will shut down. If this parameter is lowered, the demand for real-time computation will increase.

If you have problems with the connection to RSI shutting down now and then while moving the robot it is suggested to:

- Compile and install a RT-Preempt kernel for your PC.
- Give **kuka_rsi_hardware_interface** on your PC real-time priority when the RSI connection is established.

#### ros_rsi.src

This should only be edited if the start position specified within the file is not desirable for your application.

##### Copy files to controller

The files **ros_rsi.rsi** and **ros_rsi.rsi.diagram** should not be edited. All files are now ready to be copied to the Kuka controller:

1. Copy the files to a USB-stick.
2. Plug it into the teach pad or controller.
3. Log in as **Expert** or **Administrator**.
4. Copy the `ros_rsi.src` file to `KRC:\R1\Program`.
5. Copy the rest of the files to `C:\KRC\ROBOTER\Config\User\Common\SensorInterface`.

## 3. Configure the kuka_rsi_hw_interface

The **kuka_rsi_hardware_interface** needs to be configured in order to successfully communicate with RSI. Inside `/kuka_rsi_hw_interface/test` and `/kuka_rsi_hw_interface/config` in this repository is a

set of `*.yaml` files. These configuration files may be loaded into a launch-file used to start the **kuka_rsi_hardware_interface** with correct parameters, such as:

- **Joint names** corresponding to the robot description (URDF or .xacro).
- **IP address** and **port** corresponding to the RSI setup specified in **ros_rsi_ethernet.xml**.

We recommend that you copy the configuration files, edit the copies for your needs and use these files to create your own launch file. A template will be provided at a later time. However, at this time, have a look at `test_hardware_interface.launch`, `test_params.yaml`, `controller_joint_names.yaml` and `hardware_controllers.yaml` to achieve a working test-launch.

In order to successfully launch the **kuka_rsi_hardware_interface** a parameter `robot_description` needs to be present on the ROS parameter server. This parameter can be set manually or by adding this line inside the launch file (replace support package and .xacro to match your application):

```
<param name="robot_description" command="$(find xacro)/xacro.py '$(find
kuka_kr6_support)/urdf/kr6r900sixx.xacro'"/>
```

Make sure that the line is added before the `kuka_hardware_interface` itself is loaded.

## 4. Testing

At this point you are ready to test the RSI interface. Before the test, make sure that:

- You have specified the `rsi/listen_address` and `rsi/listen_port` of the **kuka_rsi_hardware_interface** to correspond with the KRL files on the controller.
- You have a launch-file loading the network parameters, robot description, kuka_hardware_interface, hardware controller and controller joint names.

The next steps describe how to launch the test file:

- In a new terminal:

```
$ roscore
```

- Open a new terminal and roslaunch the previously configured launch-file:

```
$ roslaunch kuka_rsi_hw_interface test_hardware_interface.launch sim:=false
```

The **kuka_rsi_hardware_interface** is now waiting for the robot controller to connect. Follow the next steps to connect RSI:

1. On the teach pad, enter mode **T1** for testing purposes.
2. Navigate to `KRC:\R1\Program` and select `ros_rsi.src`.

3. Press and hold an enabling switch and the run/play-button. The robot will first move to the start position.
    - A message like **Programmed path reached (BCO)** will be shown at the teach pad.
4. Press and hold again. The teach pad will post a warning **!!! Attention - Sensor correction goes active !!!**.
5. Confirm the warning and press and hold the buttons again. This time the terminal where **kuka_rsi_hardware_interface** is running should output **Got connection from robot**. The RSI connection is now up and running.
6. In a new terminal:

```
$ rosrun rqt_joint_trajectory_controller rqt_joint_trajectory_controller
```

Choose **controller manager ns** and **controller** and you should be able to move each robot joint.

- Note that T1-mode limits the robot movement velocity and is intended for testing purposes.